

An approach to improve ill-conditioned steepest descent methods, application to a parabolic optimal control problem via time domain decomposition.

Mohamed Kamel Riahi¹,

CMAI, INRIA-Saclay and X-Ecole Polytechnique, Route de Saclay, 91128 Palaiseau.

Abstract

In this paper we present a new steepest-descent type algorithm for convex optimization problems. The method combines a Newton technique together with time domain decomposition in order to achieve the optimal step-length for the given set of descent directions. This is a parallel algorithm, where the parallel tasks turn on the control during a specific time-window and turn it off elsewhere. This new technique significantly improves computational time compared with recognized methods. Convergence analysis of the algorithm is provided for an arbitrary choice of partition. Numerical experiments are presented to illustrate the efficiency of our algorithm.

Keywords: Optimal control, regularization, time-domain decomposition, Newton algorithm, convex optimization, PDEs, high performance computing, parallel algorithm.

2000 MSC: ,

1. Introduction

Typically the improvement of iterative methods is based on an implicit transformation of the original linear system in order to get a new system which has a condition number ideally close to one. This technique is known as preconditioning. Modern preconditioning techniques such as algebraic multilevel and domain decomposition methods attempt to produce efficient tools to accelerate convergence. Other techniques have introduced a different definition of the descent directions, for example, CG-method, BFGS, or its limited memory version l-BFGS. Others approaches (e.g. [1, 2, 3], without being exhaustive) propose different formulas for the line search.

The central investigation of this paper is the computation of the line search for a given set of descent directions.

Steepest descent methods [4] are usually used for solving, for example, optimization problems, control with partial differential equations (PDEs) constraints and inverse problems. Several approaches have been developed in the cases of constrained and unconstrained optimization. It is well-known that the algorithm has a slow convergence rate with ill-conditioned problems because

Email address: riahi@cmap.polytechnique.fr (Mohamed Kamel Riahi)

URL: <http://www.cmap.polytechnique.fr/~riahi> (Mohamed Kamel Riahi)

the number of iterations is proportional to the condition number of the problem. The method of J.Barzila and J.Borwein [5] based on two-point step-length for the steepest-descent method for approximating the secant equation avoids this handicap. Our method is very different, because it is based on domain decomposition that proposes block gradient descent, also because it is general where it can be coupled together with any optimization procedure.

The key idea of our method is based on a quasi-Newton technique to perform efficient real vector step-length for a set of descent directions, which is the canonical output of the time domain decomposition.

The originality of our approach consists in enabling parallel computation by considering the step-length as a vector, which achieves the optimal descent direction in a high dimensional space. The theoretical basis of our approach is presented on a simple positive definite quadratic form and applied on a more complex engineering problem involving control of system governed by PDEs. We apply the technique on a convex optimal control problem with the constrained heat equation. We test the case of well-posed problem and also the ill-posed case.

This paper is organized as follows: In Section 2, we present our method in a linear algebra framework and highlight its generality. Section 3 is devoted to the presentation of the optimal control problem with constrained PDE for both distributed and Dirichlet control problem. We present the Euler-Lagrange-system associated to the optimization problem and give the explicit formulation of the gradient in both cases. Then, we present and explain the parallel setting for the complex control problem. In Section 4, we perform the convergence analysis of the parallel algorithm. In Section 5, we present the numerical experiments that demonstrate the efficiency and the robustness of the approach. We make concluding remarks in Section 6. For completeness, we include calculus results in the Appendix.

Let Ω be a bounded domain in \mathbb{R}^3 , and $\Omega_c \subset \Omega$ a subdomain. In what follows, we denote by $\langle \cdot, \cdot \rangle_2$ (respectively $\langle \cdot, \cdot \rangle_c$) the standard $L^2(\Omega)$ (respectively $L^2(\Omega_c)$) inner-product that induces the $L^2(\Omega)$ -norm $\|\cdot\|_2$ on the domain Ω (respectively $\|\cdot\|_c$ on Ω_c). In the case of finite dimensional vector space \mathbb{R}^m , the scalar product $a^T b$ of a and b (where a^T stands for the transpose of a) is also denoted by $\langle \cdot, \cdot \rangle_2$. The scalar product with respect to the matrix A , i.e. $\langle x, Ax \rangle_2$ is denoted by $\langle x, x \rangle_A$ and the induced norm is denoted by $\|x\|_A$. We denote by $a.b$ the dot product (multiplication term by term). The Hilbert space $L^2(0, T; L^2(\Omega_c))$ will have the scalar product $\langle \cdot, \cdot \rangle_{c,T}$ that induces the norm $\|\cdot\|_{c,T}$. The transpose of the operator A is denoted by A^T .

2. Enhanced steepest descent iterations

The steepest descent algorithm minimizes at each iteration the quadratic function $q(x) = \|x - x^*\|_A^2$, where A is assumed to be an SPD matrix and x^* is the minimum of q . The vector $-\nabla q(x)$ is locally the descent direction that yields the fastest rate of decrease of the quadratic form q . Therefore all vectors of the form $x + \theta \nabla q(x)$, where θ is a suitable negative real value, decrease q . The choice of θ is found by looking for the $\min_{s < 0} q(x + s \nabla q(x))$ with the use of a line search technique. In the case where q is a quadratic form θ is given by $- \|\nabla q(x)\|_2^2 / \|\nabla q(x)\|_A^2$. We recall in **Algorithm 1** the steepest descent algorithm; *Convergence* is a boolean based on an estimate of the residual vector $r^k < \epsilon$, where ϵ is the threshold.

Our method proposes to modify the instance 5. of **Algorithm 1**. It shall consider the step-length θ as a vector of an arbitrary \hat{n} with $1 \leq \hat{n} \leq \text{size}(x)$, we shall denote this new vector as

Algorithm 1: Steepest descent.

Input: x^0 ;
1 $k = 0$;
2 while Convergence **do**
3 $r^k = \nabla q^k := \nabla q(x^k)$;
4 Compute Ar^k ;
5 Compute $\theta^k = -\|r^k\|_2^2 / \|r^k\|_A^2$;
6 $x^{k+1} = x^k + \theta^k r^k$;
7 $k = k + 1$;
8 end

$\Theta_{\hat{n}}$.

Suppose $x \in \mathbb{R}^m$. Let us introduce the partition of the identity operators: the basis $\{e_n\}_{n=1}^{\hat{n}}$ such that $\sum_{n=1}^{\hat{n}} e_n = Id$. The operators e_n are defined by $e_n : x \mapsto e_n(x) = \tilde{x}_n \in \mathbb{R}^{\frac{m}{\hat{n}}}$, where it is assumed that \hat{n} divide m with null rest. The concatenation of \tilde{x}_n for all $1 \leq n \leq \hat{n}$ is denoted by $\hat{x}_{\hat{n}} = (\tilde{x}_1, \dots, \tilde{x}_{\hat{n}})^T \in \mathbb{R}^{\frac{m}{\hat{n}} \times \hat{n}}$. Recall the gradient $\nabla = (\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_m})^T$, and define the bloc gradient $\nabla^{\hat{n}} = (\frac{\partial}{\partial \tilde{x}_1}, \dots, \frac{\partial}{\partial \tilde{x}_{\hat{n}}})^T$. In the spirit of the decomposition we investigate, in the sequel, the local descent directions as the bloc partial derivatives with respect to the bloc variables $(\tilde{x}_n)_n$. We aim, therefore, at finding $\Theta_{\hat{n}} = [\theta_1, \dots, \theta_{\hat{n}}]^T$ that ensure $\min_{(\theta_n)_n < 0} q(\hat{x}_{\hat{n}} + \Theta_{\hat{n}} \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}))$, where we recall that $\Theta_{\hat{n}} \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}})$ means the dot product.

Remark 2.1. Remark that we identify $\hat{x}_{\hat{n}}$ with $x \in \mathbb{R}^m$.

We state hereafter a motivating theorem, which proof is straightforward because the spaces are embedded.

Theorem 2.1. We denote by $q_{\hat{n}}$ the continuous functions defined by $q_{\hat{n}}(x) := q(\hat{x}_{\hat{n}})$. We have thus

$$\min_{\mathbb{R}^{\frac{m}{2\hat{n}} \times 2\hat{n}}} q_{2\hat{n}}(\hat{x}_{2\hat{n}}) \leq \min_{\mathbb{R}^{\frac{m}{\hat{n}} \times \hat{n}}} q_{\hat{n}}(\hat{x}_{\hat{n}}) \leq \min_{\mathbb{R}^{\frac{m}{2} \times 2}} q_2(\hat{x}_2) \leq \min_{\mathbb{R}^m} q_1(\hat{x}_1).$$

Furthermore, x^* the unique minimum of q satisfies

$$q_{2\hat{n}}(x^*) = q_{\hat{n}}(x^*) = q_2(x^*) = q_1(x^*),$$

where obviously $x^* = \hat{x}_{2\hat{n}}^* = \hat{x}_{\hat{n}}^* = \hat{x}_2^* = \hat{x}_1^*$.

The new algorithm we discuss in this paper proposes to define a sequence $(\hat{x}_{\hat{n}}^k)_k$ of vectors that converges to x^* (if it exists). The update formulae reads:

$$\hat{x}_{\hat{n}}^{k+1} = \hat{x}_{\hat{n}}^k + \Theta_{\hat{n}}^k \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k), \quad \forall 1 \leq n \leq \hat{n},$$

where \hat{n} is an arbitrarily chosen integer.

We shall explain now how one can accurately computes the vector step-length $\Theta_{\hat{n}}$, in the case where q is a quadratic form. In fact, denote $\Phi_{\hat{n}}(\Theta_{\hat{n}}) : \mathbb{R}^{\hat{n}} \rightarrow \mathbb{R}^{\hat{n}}$, $\Theta_{\hat{n}} \mapsto q(\hat{x}_{\hat{n}}^k + \Theta_{\hat{n}}^k \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k))$. This form is quadratic because q is. Using the chain rule, we obtain $\Phi'_{\hat{n}}(\Theta_{\hat{n}}) = \nabla^{\hat{n}} q^T(\hat{x}_{\hat{n}}^k) \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k + \Theta_{\hat{n}}^k \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k))$ and $\Phi''_{\hat{n}}(\Theta_{\hat{n}}) = \nabla^{\hat{n}} q^T(\hat{x}_{\hat{n}}^k) \cdot (\nabla^{\hat{n}})^2 q(\hat{x}_{\hat{n}}^k + \Theta_{\hat{n}}^k \cdot \nabla q(\hat{x}_{\hat{n}}^k) \cdot \nabla q(\hat{x}_{\hat{n}}^k))$. Here $(\nabla^{\hat{n}})^2 = \nabla^{\hat{n}} \cdot (\nabla^{\hat{n}})^T$ is

the Hessian matrix. We thus have the Taylor expansion $\Phi_{\hat{n}}(\Theta_{\hat{n}}^k) = \Phi_{\hat{n}}(\mathbf{0}^T) + (\Theta_{\hat{n}}^k)^T \Phi_{\hat{n}}'(\mathbf{0}^T) + \frac{1}{2}(\Theta_{\hat{n}}^k)^T \Phi_{\hat{n}}''(\mathbf{0}^T) \Theta_{\hat{n}}^k$, with $\mathbf{0}^T := (0, \dots, 0)^T \in \mathbb{R}^{\hat{n}}$. Then the vector $\Theta_{\hat{n}}^k$ that annuls the gradient writes:

$$\Theta_{\hat{n}}^k = -(\nabla^{\hat{n}} q^T(\hat{x}_{\hat{n}}^k) \cdot (\nabla^{\hat{n}})^2 q(\hat{x}_{\hat{n}}^k) \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k))^{-1} (\nabla^{\hat{n}} q^T(\hat{x}_{\hat{n}}^k) \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k))^T. \quad (1)$$

Algorithm 1, therefore, has a bloc structure which can be solved in parallel. This is due to the fact that partial derivatives can be computed independently. The new algorithm is thus as follows

Algorithm 2: Enhanced steepest descent.

```

k = 0;
Input:  $\hat{x}_{\hat{n}}^0$ ;
1 while Convergence do
2   forall the  $1 \leq n \leq \hat{n}$  do
3      $\hat{x}_n^k = e_n(\hat{x}_{\hat{n}}^k)$ ;
4      $r_n = \frac{\partial}{\partial \hat{x}_n^k} q(\hat{x}_{\hat{n}}^k)$ ;
5   end
6   Concatenate  $\hat{r}_{\hat{n}} = [r_1, \dots, r_{\hat{n}}]^T$ ;
7   Assemble the vector  $D_{\hat{n}} = \hat{r}_{\hat{n}}^T \cdot \hat{r}_{\hat{n}}$ ;
8   Assemble the matrix  $H_{\hat{n}} = \hat{r}_{\hat{n}} \cdot A \cdot \hat{r}_{\hat{n}}$ ;
9   Compute  $\Theta_{\hat{n}}^k = -H_{\hat{n}}^{-1} D_{\hat{n}}$ ;
10   $\hat{x}_{\hat{n}}^{k+1} = \hat{x}_{\hat{n}}^k + \Theta_{\hat{n}}^k \cdot \nabla^{\hat{n}} q(\hat{x}_{\hat{n}}^k)$ ;
11  k = k + 1;
12 end

```

3. Application to a parabolic optimal control problem

In this part we are interested in the application of **Algorithm 2** in a complicated computational engineering problem, involving optimization with constrained PDE. In particular, we deal with the optimal control problem of a system, which is governed by the heat equation. We shall present two types of control problems. The first concerns the distributed optimal control and the second concerns the Dirichlet boundary control.

3.1. Distributed control problem

Let us briefly present the steepest descent method applied to the following optimal control problem: find v^* such that

$$J(v^*) = \min_{v \in L^2(0,T;L^2(\Omega_c))} J(v), \quad (2)$$

where J is a quadratic cost functional defined by

$$J(v) = \frac{1}{2} \|y(T) - y^{target}\|_2^2 + \frac{\alpha}{2} \int_I \|v\|_c^2 dt, \quad (3)$$

where y^{target} is a given target state and $y(T)$ is the state variable at time $T > 0$ of the heat equation controlled by the variable v over $I := [0, T]$. The Tikhonov regularization parameter α is also introduced to penalize the control's L^2 -norm over the time interval I , this quantity is also called energy.

The optimality system of our problem reads:

$$\begin{cases} \partial_t y - \sigma \Delta y = \mathcal{B}v, & \text{on } I \times \Omega, \\ y(t=0) = y_0. \end{cases} \quad (4)$$

$$\begin{cases} \partial_t p + \sigma \Delta p = 0, & \text{on } I \times \Omega, \\ p(t=T) = y(T) - y^T. \end{cases} \quad (5)$$

$$\nabla J(v) = \alpha v + \mathcal{B}^T p = 0, \text{ on } I \times \Omega. \quad (6)$$

In the above equations, the operator \mathcal{B} is a linear operator that distributes the control in Ω_c .

3.2. Dirichlet boundary control problem

In this subsection we are concerned with the PDE constrained Dirichlet boundary optimal control problem, where we aim at minimizing the cost functional J_Γ defined by

$$J_\Gamma(v_\Gamma) = \frac{1}{2} \|y(T) - y^{target}\|_2^2 + \frac{\alpha}{2} \int_I \|v_\Gamma\|_\Gamma^2 dt, \quad (7)$$

using only a boundary control on $\Gamma \subset \partial\Omega$. The involved optimality system reads

$$\begin{cases} \partial_t y_\Gamma - \sigma \Delta y_\Gamma &= f & \text{on } I \times \Omega \\ y_\Gamma &= v_\Gamma & \text{on } I \times \Gamma \\ y_\Gamma &= g & \text{on } I \times \{\partial\Omega \setminus \Gamma\} \\ y_\Gamma(0) &= y_0. \end{cases} \quad (8)$$

$$\begin{cases} -\partial_t p_\Gamma - \sigma \Delta p_\Gamma &= 0 & \text{on } I \times \Omega \\ p_\Gamma &= 0 & \text{on } I \times \partial\Omega \\ p_\Gamma(T) &= y_\Gamma(T) - y^{target}. \end{cases} \quad (9)$$

$$\nabla J_\Gamma(v_\Gamma) = \alpha v_\Gamma - \nabla p_\Gamma \cdot \vec{n} = 0 \quad \text{on } I \times \Gamma \quad (10)$$

where $f \in L^2(\Omega)$ is any source term, $g \in L^2(\Gamma)$ and \vec{n} is the outward unit normal on Γ .

3.3. Steepest descent algorithm for optimal control of constrained PDE

We propose to solve the optimization problem iteratively. Let us denote by k the current iteration superscript. We suppose that v^0 is known. The first order steepest descent algorithm updates the control variable as follows:

$$v^k = v^{k-1} + \theta^{k-1} \nabla J(v^{k-1}), \text{ for } k \geq 1. \quad (11)$$

The step-length $\theta^{k-1} \in \mathbb{R}^+ \setminus \{0\}$ in the direction of the gradient $\nabla J(v^{k-1}) = \{\alpha v^{k-1} + \mathcal{B}^T p^{k-1}, \alpha v_\Gamma - \nabla p_\Gamma \cdot \vec{n}\}$ is computed as follows:

$$\theta^{k-1} = -\|\nabla J(v^{k-1})\|_{c,I}^2 / \|\nabla J(v^{k-1})\|_{\nabla^2 J}^2.$$

This step-length is optimal in the sense that it minimizes the functional $\theta \rightarrow J(v^{k-1} + \theta \nabla J(v^{k-1}))$ (see e.g. [6]). The rate of convergence of this technique is $(\frac{\kappa-1}{\kappa+1})^2$, where κ is the condition number of the quadratic form, namely the Hessian of the cost functional J .

3.4. Time-domain decomposition algorithm

Consider \hat{n} subdivisions of the time interval $I = \cup_{n=1}^{\hat{n}} I_n$, consider also the following convex cost functional J :

$$J(v_1, v_2, \dots, v_{\hat{n}}) = \frac{1}{2} \|\mathcal{Y}(T) - y^{target}\|_2^2 + \frac{\alpha}{2} \sum_{n=1}^{\hat{n}} \int_{I_n} \|v_n\|_c^2 dt, \quad (12)$$

where $v_n, n = 1, \dots, \hat{n}$ are control variables with time support included in $I_n, n = 1, \dots, \hat{n}$. The state $\mathcal{Y}(T)$ depends on v ; the concatenation of controls $v_1, v_2, \dots, v_{\hat{n}}$. Let us define $\Theta_{\hat{n}} := (\theta_1, \theta_2, \dots, \theta_{\hat{n}})^T$ where $\theta_n \in \mathbb{R} \setminus \{0\}$. For any admissible control $w = (w_1, \dots, w_{\hat{n}})^T$ we also define $\varphi_{\hat{n}}(\Theta_{\hat{n}}) := J(v + \sum_{n=1}^{\hat{n}} \theta_n w_n)$, which is quadratic. We have:

$$\varphi_{\hat{n}}(\Theta_{\hat{n}}) = \varphi_{\hat{n}}(\mathbf{0}) + \Theta_{\hat{n}}^T \nabla \varphi_{\hat{n}}(\mathbf{0}) + \frac{1}{2} \Theta_{\hat{n}}^T \nabla^2 \varphi_{\hat{n}}(\mathbf{0}) \Theta_{\hat{n}}, \quad (13)$$

where $\mathbf{0} = (0, \dots, 0)^T$. Therefore we can write $\nabla \varphi_{\hat{n}}(\Theta_{\hat{n}}) \in \mathbb{R}^{\hat{n}}$ as $\nabla \varphi_{\hat{n}}(\Theta_{\hat{n}}) = D(v, w) + H(v, w) \Theta_{\hat{n}}$, where the Jacobian vector and the Hessian matrix are given respectively by:

$$\begin{aligned} D(v, w) &:= (\langle \nabla J(v), e_1(w) \rangle_c, \dots, \langle \nabla J(v), e_{\hat{n}}(w) \rangle_c)^T \in \mathbb{R}^{\hat{n}}, \\ H(v, w) &:= (H_{n,m})_{n,m}, \text{ for } H_{n,m} = \langle e_n(w), e_m(w) \rangle_{\nabla^2 J}. \end{aligned}$$

Here, $(e_n)_n$ are Heaviside functions with support on the time interval I_n .

The solution $\Theta_{\hat{n}}^*$ of $\nabla \varphi_{\hat{n}}(\Theta_{\hat{n}}) = \mathbf{0}$ can be written in the form:

$$\Theta_{\hat{n}}^* = -H^{-1}(v, w) D(v, w). \quad (14)$$

Enlarging the domain of minimization guarantees a lower minimum. Since $(v_n + \theta_n w_n)_n$ are controls with disjoint time-support, they drive independent solutions. In fact, the processors require only the scalar θ_n in order to perform the update. This feature is very important because it reduces the communication of a vector $\theta_n w_n$ to the communication of a scalar θ_n .

In the parallel distributed control problem, we are concerned with the following optimality system:

$$\begin{cases} \partial_t \mathcal{Y}_n - \sigma \Delta \mathcal{Y}_n = \mathcal{B} v_n, & \text{on } I \times \Omega, \\ \mathcal{Y}_n(t=0) = \delta_n^0 y_0. \end{cases} \quad (15)$$

$$\mathcal{Y}(T) = \sum_{n=1}^{\hat{n}} \mathcal{Y}_n(T) \quad (16)$$

$$\begin{cases} \partial_t \mathcal{P} + \sigma \Delta \mathcal{P} = 0, & \text{on } I \times \Omega, \\ \mathcal{P}(t=T) = \mathcal{Y}(T) - y^{target}. \end{cases} \quad (17)$$

$$\nabla J\left(\sum_{n=1}^{\hat{n}} v_n\right) = \mathcal{B}^T \mathcal{P} + \alpha \sum_{n=1}^{\hat{n}} v_n = 0, \text{ on } I \times \Omega. \quad (18)$$

and for the Dirichlet control problem we are concerned with:

$$\begin{cases} \partial_t \mathcal{Y}_{n,\Gamma} - \sigma \Delta \mathcal{Y}_{n,\Gamma} &= f & \text{on } I \times \Omega \\ \mathcal{Y}_{n,\Gamma} &= v_{n,\Gamma} & \text{on } I \times \Gamma \\ \mathcal{Y}_{n,\Gamma} &= g & \text{on } I \times \{\partial\Omega \setminus \Gamma\} \\ \mathcal{Y}_{n,\Gamma}(0) &= \delta_n^0 y_0. \end{cases} \quad (19)$$

$$\mathcal{Y}_\Gamma(T) = \sum_{n=1}^{\hat{n}} \mathcal{Y}_{n,\Gamma}(T) \quad (20)$$

$$\begin{cases} -\partial_t \mathcal{P}_\Gamma - \sigma \Delta \mathcal{P}_\Gamma &= 0 & \text{on } I \times \Omega \\ \mathcal{P}_\Gamma &= 0 & \text{on } I \times \partial\Omega \\ \mathcal{P}_\Gamma(T) &= \mathcal{Y}_\Gamma(T) - y^{target}. \end{cases} \quad (21)$$

$$\nabla J_\Gamma\left(\sum_{n=1}^{\hat{n}} v_{n,\Gamma}\right) = -(\nabla \mathcal{P}_\Gamma)^T \vec{n} + \alpha \sum_{n=1}^{\hat{n}} v_{n,\Gamma} = 0 \quad \text{on } I \times \Gamma. \quad (22)$$

The resolution of Eqs. (15) and (19) is fully performed in parallel over I . As before the superscript k denotes the iteration index for the new algorithm as well. The update formulae for the control variable v^k is given by:

$$v_n^k = v_n^{k-1} + \theta_n^{k-1} \begin{cases} \mathcal{B}^T \mathcal{P}^{k-1} + \alpha \sum_{n=1}^{\hat{n}} v_n^{k-1} \\ -(\nabla \mathcal{P}_\Gamma^{k-1})^T \vec{n} + \alpha \sum_{n=1}^{\hat{n}} v_{n,\Gamma}^{k-1}. \end{cases}$$

We show hereafter how to assemble vector step-length Θ_n^k at each iteration. For the purposes of notation we denote by H_k the k -th iteration of the Hessian matrix $H(\nabla J(v^k), \nabla J(v^k))$ and by D_k the k -th iteration of the Jacobian vector $D(\nabla J(v^k), \nabla J(v^k))$.

Line search is performed with quasi-Newton techniques that uses at each iteration k a Hessian matrix H_k and Jacobian vector D_k defined respectively by:

$$D_k := \left(\langle \nabla J(v^k), e_1(\nabla J(v^k)) \rangle_c, \dots, \langle \nabla J(v^k), e_{\hat{n}}(\nabla J(v^k)) \rangle_c \right)^T, \quad (23)$$

$$(H_k)_{n,m} := \langle e_n(\nabla J(v^k)), e_m(\nabla J(v^k)) \rangle_{\nabla^2 J}. \quad (24)$$

We denote the condition number of the Hessian matrix $\nabla^2 J$ as: $\kappa := \kappa(\nabla^2 J) := \lambda_{\max} \lambda_{\min}^{-1}$, with $\lambda_{\max} := \lambda_{\max}(\nabla^2 J)$ the largest eigenvalue of $\nabla^2 J$ and $\lambda_{\min} := \lambda_{\min}(\nabla^2 J)$ its smallest eigenvalue.

According to Eq.(14) we have

$$\Theta_n^k = -H_k^{-1} D_k. \quad (25)$$

From Eq.(13) we have:

$$J(v^{k+1}) = J(v^k) + (\Theta_n^k)^T D_k + \frac{1}{2} (\Theta_n^k)^T H(v, w) \Theta_n^k. \quad (26)$$

Our parallel algorithm to minimize the cost functional (see Eq.(12)) is stated as follows:

Since $(v_n)_n$ have disjoint time-support, thanks to the linearity, the notation $e_n(\nabla J(v^k))$ is nothing but $\nabla J(v_n^k)$, where v^k is the concatenation of $v_1^k, \dots, v_{\hat{n}}^k$.

In **Algorithm 3** instances 6,7,9,10 and 11 are trivial tasks in regards to computational effort.

Algorithm 3: Enhanced steepest descent algorithm for the optimal control problem.

```

0 Input:
1 while Convergence do
2   forall the  $1 \leq n \leq \hat{n}$  do
3     Solve  $\mathcal{Y}_n(T)(v_n^k)$  of Eq.(15)(or Eq.(19)) in parallel for all  $1 \leq n \leq \hat{n}$ ;
4     Compute  $(D_k)_n$  of Eq.(23) in parallel for all  $1 \leq n \leq \hat{n}$ ;
5   end
6   Gather  $(D_k)_n$  from processor  $n$ ,  $2 \leq n \leq \hat{n}$  to master processor;
7   Assemble the Hessian matrix  $H_k$  according to Eq.(24) with master processor;
8   Compute  $\mathcal{P}(t)$  with the backward problem according to Eq.(17) (or Eq.(21)) ;
9   Compute the inversion of the SPD matrix  $H_k$  and calculate  $\Theta_n^k$  using Eq.(25);
10  Broadcast  $\theta_n^k$  from master processor to all the other  $(\hat{n} - 1)$  processors;
11  Update local control variable  $v_n^{k+1}$  in parallel as :

      
$$v_n^{k+1} = v_n^k + \theta_n^k e_n(\nabla J(v^k)) \quad \text{for all } 1 \leq n \leq \hat{n},$$


    and go to step 2;
12   $k = k + 1$ ;
13 end

```

4. Convergence analysis

This section provides the proof of convergence of **Algorithm 3**. The size of the matrix H_k is bounded above by the number of the time step discretizations τ . We have thus $\hat{n} \leq \frac{T}{\tau}$ so we may use direct solver in order to obtain an exact inversion.

In the sequel, we suppose that $\|\nabla J(v^k)\|_c$ does not vanish; otherwise the algorithm has already converged.

prop 4.1. *The increase in the cost functional between two successive controls v^k and v^{k+1} is bounded below by:*

$$J(v^k) - J(v^{k+1}) \geq \frac{1}{2\kappa(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|_{\nabla^2 J}^2}. \quad (27)$$

Proof. Using Eq.(25) and Eq.(26), we can write:

$$J(v^k) - J(v^{k+1}) = \frac{1}{2} D_k^T H_k^{-1} D_k. \quad (28)$$

Preliminaries: From the definition of the Jacobian vector D_k we have

$$\begin{aligned}
\|D_k\|_2^2 &= \sum_{n=1}^{\hat{n}} \langle \nabla J(v^k), e_n(\nabla J(v^k)) \rangle_c^2, \\
&= \sum_{n=1}^{\hat{n}} \langle e_n(\nabla J(v^k)), e_n(\nabla J(v^k)) \rangle_c^2, \\
&= \sum_{n=1}^{\hat{n}} \|e_n(\nabla J(v^k))\|_c^4, \\
&= \|\nabla J(v^k)\|_c^4.
\end{aligned}$$

Furthermore since H_k is an SPD matrix we have

$$\lambda_{\min}(H_k^{-1}) = \frac{1}{\lambda_{\max}(H_k)},$$

from which we deduce:

$$\frac{1}{\lambda_{\min}(H_k)} \geq \frac{1}{\frac{1}{\hat{n}} 1_{\hat{n}}^T H_k 1_{\hat{n}}}.$$

Moreover, we have:

$$\begin{aligned}
D_k^T H_k^{-1} D_k &= \frac{D_k^T H_k^{-1} D_k}{\|D_k\|_2^2} \|D_k\|_2^2 \geq \lambda_{\min}(H_k^{-1}) \|D_k\|_2^2 \\
&= \lambda_{\min}(H_k^{-1}) \lambda_{\min}(H_k) \frac{\|\nabla J(v^k)\|_c^4}{\lambda_{\min}(H_k)} \\
&\geq \frac{\lambda_{\min}(H_k)}{\lambda_{\max}(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\frac{1}{\hat{n}} 1_{\hat{n}}^T H_k 1_{\hat{n}}} \\
&= \frac{\hat{n}}{\kappa(H_k)} \|\nabla J(v^k)\|_{\nabla^2 J}^{-2} \|\nabla J(v^k)\|_c^4.
\end{aligned}$$

Since the partition number \hat{n} is greater than or equal to 1, we conclude that :

$$D_k^T H_k^{-1} D_k \geq \frac{\|\nabla J(v^k)\|_{\nabla^2 J}^{-2} \|\nabla J(v^k)\|_c^4}{\kappa(H_k)}. \quad (29)$$

Hence, using Eq.(28) we get the stated result. \square

\square

Theorem 4.2. *The control sequence $(v^k)_{k \geq 1}$ of **Algorithm 1** converges with any partition \hat{n} of sub intervals. Furthermore we have:*

$$\|v^k - v^{\star}\|_{\nabla^2 J}^2 \leq r^k \|v^0 - v^{\star}\|_{\nabla^2 J}^2,$$

where the rate of convergence $r := \left(1 - \frac{4\kappa}{\kappa(H_k)(\kappa+1)^2}\right)$ satisfies $0 \leq r < 1$.

Proof. We denote by v^\star the optimal control that minimizes J . The equality

$$J(v) = J(v^\star) + \frac{1}{2} \langle v - v^\star, v - v^\star \rangle_{\nabla^2 J} = J(v^\star) + \frac{1}{2} \|v - v^\star\|_{\nabla^2 J}^2,$$

holds for any control v ; in particular we have:

$$\begin{aligned} J(v^{k+1}) &= J(v^\star) + \frac{1}{2} \|v^{k+1} - v^\star\|_{\nabla^2 J}^2, \\ J(v^k) &= J(v^\star) + \frac{1}{2} \|v^k - v^\star\|_{\nabla^2 J}^2. \end{aligned}$$

Consequently, by subtracting the equations above, we obtain

$$J(v^{k+1}) - J(v^k) = \frac{1}{2} \|v^{k+1} - v^\star\|_{\nabla^2 J}^2 - \frac{1}{2} \|v^k - v^\star\|_{\nabla^2 J}^2. \quad (30)$$

Since J is quadratic, we have $\nabla^2 J(v^k - v^\star) = \nabla^2 J(v^k)$, that is $v^k - v^\star = (\nabla^2 J)^{-1} \nabla J(v^k)$. Therefore we deduce:

$$\begin{aligned} \|v^k - v^\star\|_{\nabla^2 J}^2 &= \langle v^k - v^\star, v^k - v^\star \rangle_{\nabla^2 J} \\ &= \langle v^k - v^\star, \nabla^2 J, v^k - v^\star \rangle_c \\ &= \langle (\nabla^2 J)^{-1} \nabla J(v^k), \nabla^2 J, (\nabla^2 J)^{-1} \nabla J(v^k) \rangle_c \\ &= \langle \nabla J(v^k), (\nabla^2 J)^{-1}, \nabla J(v^k) \rangle_c \\ &= \|\nabla J(v^k)\|_{(\nabla^2 J)^{-1}}^2. \end{aligned} \quad (31)$$

Because of Eq.(28), we also have

$$J(v^{k+1}) - J(v^k) = -\frac{1}{2} D_k^T H_k^{-1} D_k.$$

Using Eq.(30) and the above, we find that:

$$\|v^{k+1} - v^\star\|_{\nabla^2 J}^2 = \|v^k - v^\star\|_{\nabla^2 J}^2 - D_k^T H_k^{-1} D_k.$$

Moreover, according to Eqs (29)-(31), we obtain the following upper bound:

$$\begin{aligned} \|v^{k+1} - v^\star\|_{\nabla^2 J}^2 &\leq \|v^k - v^\star\|_{\nabla^2 J}^2 - \frac{1}{\kappa(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|_{\nabla^2 J}^2} \\ &\leq \|v^k - v^\star\|_{\nabla^2 J}^2 \left(1 - \frac{1}{\kappa(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|_{\nabla^2 J}^2 \|\nabla J(v^k)\|_{(\nabla^2 J)^{-1}}^2} \right). \end{aligned} \quad (32)$$

Using the Kantorovich inequality [7, 8] (see also The Appendix) :

$$\frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|_{\nabla^2 J}^2 \|\nabla J(v^k)\|_{(\nabla^2 J)^{-1}}^2} \geq \frac{4\lambda_{\max}\lambda_{\min}}{(\lambda_{\max} + \lambda_{\min})^2}. \quad (33)$$

Then

$$1 - \frac{1}{\kappa(H_k)} \frac{\|\nabla J(v^k)\|_c^4}{\|\nabla J(v^k)\|_{\nabla^2 J}^2 \|\nabla J(v^k)\|_{(\nabla^2 J)^{-1}}^2} \leq 1 - \frac{4\underline{\kappa}}{\kappa(H_k)(\underline{\kappa} + 1)^2}.$$

Finally we obtain the desired results for any partition to \hat{n} subdivision, namely

$$\|v^k - v^*\|_{\nabla^2 J}^2 \leq \left(1 - \frac{4\underline{\kappa}}{\kappa(H_k)(\underline{\kappa} + 1)^2}\right)^k \|v^0 - v^*\|_{\nabla^2 J}^2.$$

The proof is therefore complete. \square \square

Remark that for $\hat{n} = 1$, we immediately get $\kappa(H_k) = 1$ and we recognize the serial steepest gradient method, which has convergence rate $\left(\frac{\underline{\kappa}-1}{\underline{\kappa}+1}\right)^2$. The iterative procedure is therefore very slow as $\underline{\kappa}$ rises. We present in what follows numerical results that demonstrated the efficiency of our algorithm, even the theoretical proof of the convergence rate is not optimal with respect to \hat{n} , because of the difficulty to estimate $\kappa(H_k)$ which depends on the set of the descent directions and the problem it self. Tests consider examples of well-posed and ill-posed control problem.

5. Numerical experiments

In order to validate the method in a linear algebra framework, we first implement a simple code with the scientific programming language Scilab [9]. The implemented algorithm considers the minimization of a quadratic form as we shall explain in the sequel.

5.1. Simple linear algebra program

Consider A an SPD m -by- m matrix and a real vector $b \in \mathbb{R}^m \cap \text{rank}(A)$. We aim at solving iteratively the linear system $Ax = b$, by minimizing the following quadratic form

$$q(x) = \frac{1}{2}x^T Ax - x^T b. \quad (34)$$

In the following, we denote by \hat{n} the partition number of the unknown $x \in \mathbb{R}^m$. Without loss of generality, the partition is supposed to be uniform; it is thus assumed that \hat{n} divides m with a null rest.

We give in Fig. 1 a SCILAB function that builds the vector step-length $\Theta_{\hat{n}}^k$ as stated in Eq. (1). Several tests has been carried out. For each test, we randomly generate an SPD sparse matrix $A = \gamma mI + R$, where $\gamma > 1$, I is the identity matrix and R is a symmetric random matrix. This way the matrix A is diagonally dominant, hence SPD. For each matrix A we proceed to minimize the quadratic form defined in Eq.(34) with various subdivisions on \hat{n} . In the case of a rapidly vanishing Eigen values of A we use a Tikhonov regularization. In this case A becomes $A = A + \alpha I$. This technique helps us to manipulate the condition number and the coercivity of the handled problem.

Fig. 2 presents the improvement quality of the algorithm against the serial case $\hat{n} = 1$. In the left side of Fig. 2 we present the cost function decay versus the iteration number of the algorithm. Several choices of partition on \hat{n} are carried out. In the right side of the Fig. 2 we give the logarithmic representation of the relative error $\frac{\|x^k - x^*\|_2}{\|x^*\|_2}$, where x^* is the exact solution of the linear system.

```

1 function [P]=Build_Hk(n,A,b,xk,dJk)
2 m=size(A,1); l=m/n; ii=modulo(m,n);
3 if ii~=0 then
4     printf("Bad choice of partition: n !");
5     printf("Please chose an other n!");
6     abort;
7 end
8 dJkn=zeros(m,n); Dk=[];
9 for i=1:n
10     dJkn((i-1)*l+1:i*l,i)= -dJk((i-1)*l+1:i*l);
11     Dk(i)=dJkn(:,i)'*(A*xk-b);
12 end
13 Hk=[];
14 for i=1:n
15     for j=i:n
16         Hktmp=A*dJkn(:,j);
17         Hk(i,j)=dJkn(:,i)'*Hktmp;
18         Hk(j,i)=Hk(i,j);
19     end
20 end
21 theta=-Hk\Dk;
22 P = eye(m,m);
23 for i=1:n
24     P((i-1)*l+1:i*l,(i-1)*l+1:i*l)=theta(i).*eye(l,l);
25 end
26 endfunction

```

Figure 1: Scilab function to build the vector step length, for the linear algebra program.

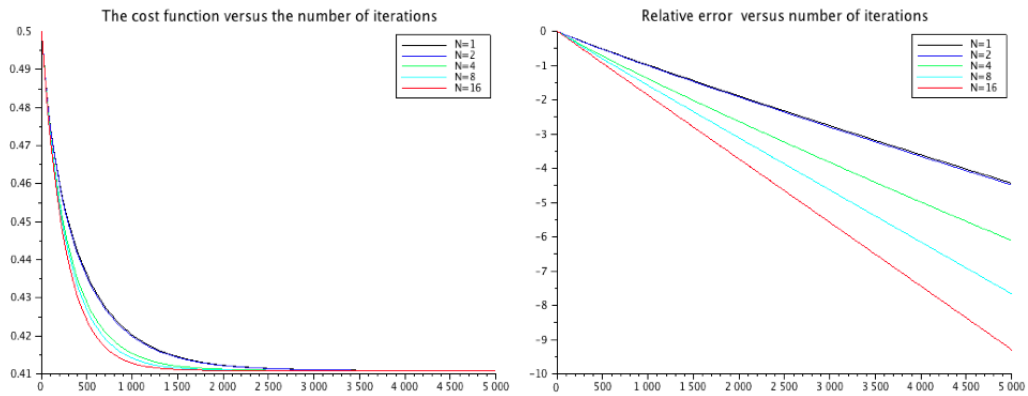


Figure 2: Cost functional decay versus time of computation for several values of \hat{n} (i.e. the number of processors used), results from the linear algebra Scilab program.

5.2. Finite element program

We discuss in the following section the application of our algorithm for the optimal control of a system governed by the heat equation. This is a typical optimal control problem faced by engineers.

Our tests deal with the 2D-heat equation on the bounded domain $\Omega = [0, 1] \times [0, 1]$. We consider, three types of test problems on both distributed and Dirichlet controls. Tests vary according to the theoretical difficulty of the control problem [10, 11, 12]. Indeed, with the minimization problem of the quadratic cost functional, we vary the regularization parameter α and also change the initial and target solutions in order to handle more severe control problems as has been tested in [10].

As has already been presented in the previous section, the optimal control problem is regularized in the sense of Tikhonov, in order to ensure on one hand well-posedness and on the other hand to penalize the control on the cost function. Numerical tests concern also the case where the regularizing parameter α tends to zero. In this case, the optimal control problem becomes an approximated controllability problem in the sense that it tries to reach, as close as it can, the target solution. With this strategy, we accentuate the ill-posedness degree of the handled problem. We also consider improper-posed problems in the case of controllability approximation, where the target solution doesn't belong to the space of the reachable solutions.

In order to emphasize the role of the parameter α in the problems, they are tagged as \mathcal{P}_i^α where the index i refers to the problem $\{1, 2, 3, 4\}$.

Tests that concern the distributed control problem are produced with control that acts on $\Omega_c \subset \Omega$, with $\Omega_c = [0, \frac{1}{3}] \times [0, \frac{1}{3}]$, whereas Dirichlet boundary control problem only concerns $\Gamma \subset \partial\Omega$, with $\Gamma = \{(x_1, x_2) \in \partial\Omega, |x_2 = 0\}$. The time horizon of the problem is fixed to $T = 6.4$ and the small time step is $\tau = 0.01$. In order to have a better control of the time evolution we put the diffusion coefficient $\sigma = 0.01$.

We suppose from now on that the computational domain Ω is a polygonal domain of the plane \mathbb{R}^2 . We then introduce a triangulation \mathcal{T}_h of Ω ; the subscript h stands for the largest length of the edges of the triangles that constitute \mathcal{T}_h . The solution of the heat equation at a given time t belongs to $H^1(\Omega)$. The source terms and other variables are elements of $L^2(\Omega)$. Those infinite dimensional spaces are therefore approximated with the finite-dimensional space V_h , characterized by \mathbb{P}_1 the space of the polynomials of degree ≤ 1 in two variables (x_1, x_2) . We have $V_h := \{u_h | u_h \in C^0(\bar{\Omega}), u_{h_K} \in \mathbb{P}_1, \text{ for all } K \in \mathcal{T}_h\}$. In addition, Dirichlet boundary conditions (where the solution is in $H_0^1(\Omega)$ i.e. vanishing on boundary $\partial\Omega$) are taken into account via penalization of the vertices on the boundaries. The time dependence of the solution is approximated via the implicit Euler scheme. The inversion operations of matrices is performed by the UMFPACK solver. We use the trapezoidal method in order to approximate integrals defined on the time interval.

The numerical experiments were run over a parallel machine with 24 CPU's AMD with 800 MHz in a Linux environment. We code two FreeFem++ [13] scripts for the distributed and Dirichlet control. We use MPI library in order to achieve parallelism.

5.2.1. First test problem

we consider a well-posed optimal control problem [14] on the heat equation. The control is considered first to be distributed and then Dirichlet. For the distributed optimal control problem we first use the functions

$$\begin{aligned} y_0(x_1, x_2) &= \exp(-\gamma 2\pi((x_1 - .7)^2 + (x_2 - .7)^2)) \\ y^{target}(x_1, x_2) &= \exp(-\gamma 2\pi((x_1 - .3)^2 + (x_2 - .3)^2)), \end{aligned} \quad (\mathcal{P}_1^\alpha)$$

as initial condition and target solution respectively. The real valued γ is introduced to force the Gaussian to have support strictly included in the domain and verify the boundary conditions. We look for the minimizer of a cost functional of type Eq. (3). The decay of the cost function with

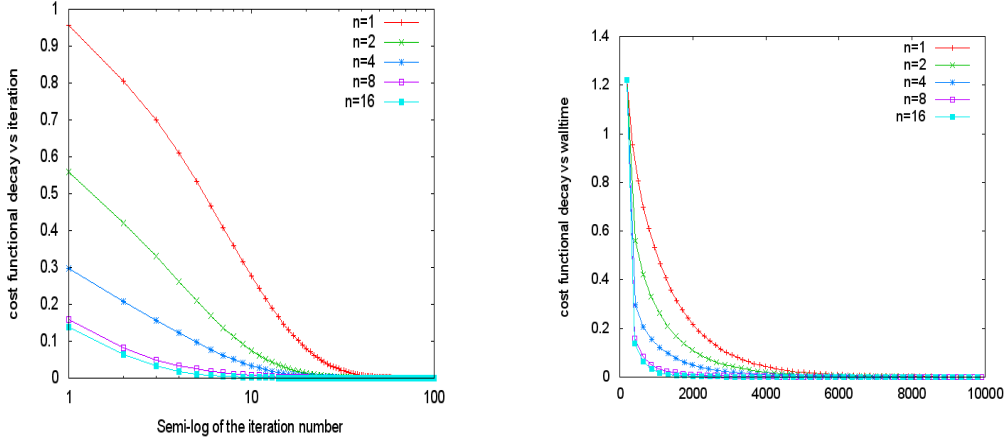


Figure 3: First test problem, for \mathcal{P}_1^α : Normalized and shifted cost functional values versus iteration number (left) and versus computational time (right) for several values of \hat{n} (i.e. the number of processors used).

respect to the iterations of our algorithm is presented in Fig. 3 on the left side, and the same results are given with respect to the computational time on the right side. We show that the algorithm accelerates with respect to the partition number \hat{n} and also preserves the accuracy of the serial resolution (i.e. $\hat{n} = 1$) in the sense that all tests, independently of \hat{n} , always converge to the unique solution. This is in agreement with Theorem (4.2), which proves the convergence of the algorithm to the optimal control (if it exists [14]) for an arbitrary partition choice \hat{n} .

We test a second problem with an a priori known solution of the heat equation. The considered problem has

$$\begin{aligned} y_0(x_1, x_2) &= \sin(\pi x_1) \sin(\pi x_2) \\ y^{target}(x_1, x_2) &= \exp(-2\pi^2 \sigma T) \sin(\pi x_1) \sin(\pi x_2), \end{aligned} \quad (\mathcal{P}_2^\alpha)$$

as initial condition and target solution respectively. Remark that the target solution is taken as a solution of the heat equation at time T . The results of this test are presented in Fig. 5, which shows the decay in values of the cost functional versus the iterations of the algorithm on the left side and versus the computational time on the right side.

We give in Fig. 4 and Fig. 6 several rows value snapshots (varying the \hat{n}) of the control and its corresponding controlled final solution $y(T)$. Notice the stability and the accuracy of the method with any choice of \hat{n} .

For the Dirichlet boundary control problem we choose the following functions as source term, initial condition and target solution:

$$\begin{aligned} f(x_1, x_2, t) &= 3\pi^3 \sigma \exp(2\pi^2 \sigma t) (\sin(\pi x_1) + \sin(\pi x_2)) \\ y_0(x_1, x_2) &= \pi (\sin(\pi x_1) + \sin(\pi x_2)) \\ y^{target}(x_1, x_2) &= \pi \exp(2\pi^2 \sigma) (\sin(\pi x_1) + \sin(\pi x_2)), \end{aligned} \quad (\mathcal{P}_3^\alpha)$$

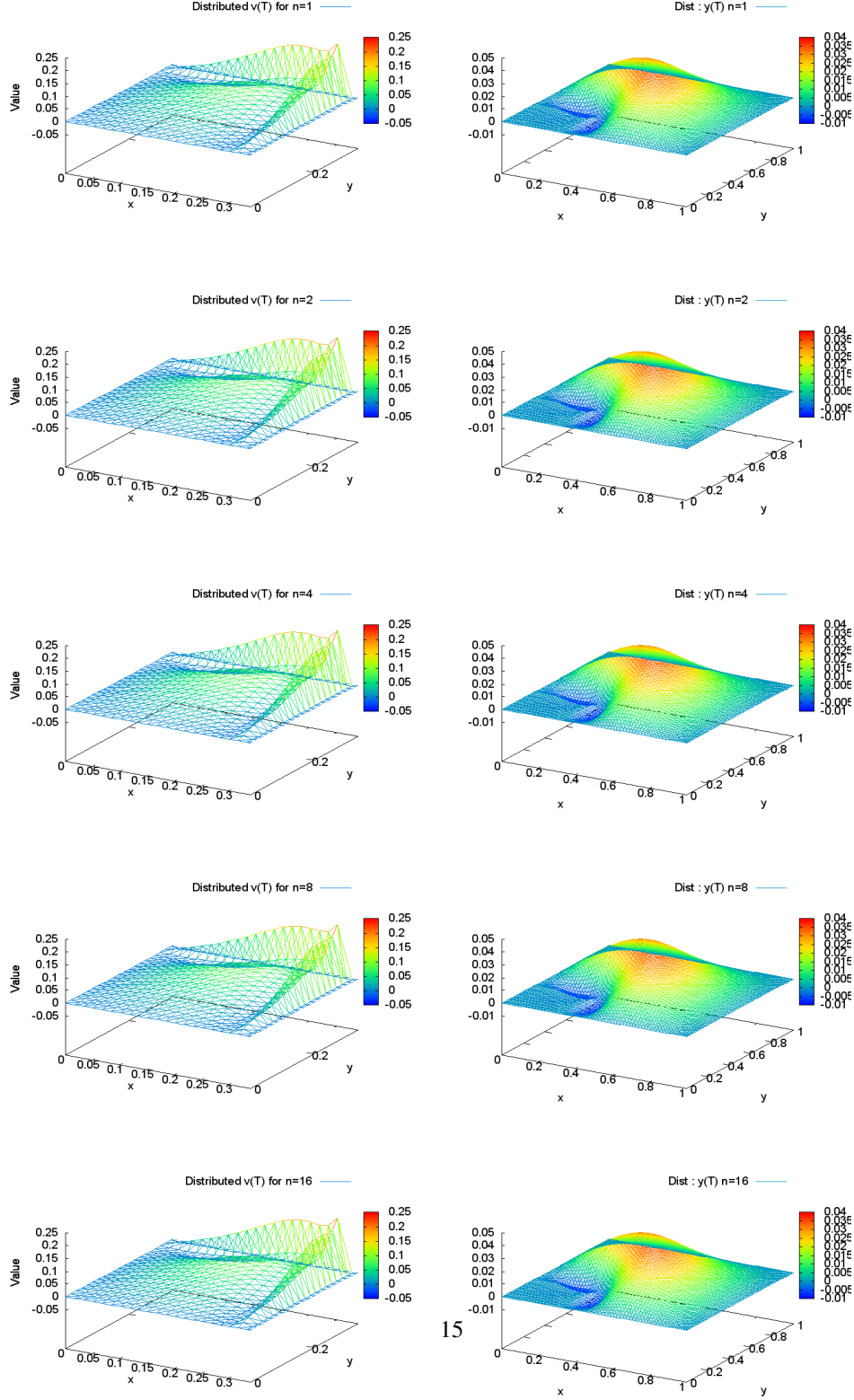


Figure 4: Several rows value snapshots in \hat{n} of the distributed optimal control on the left column and its corresponding controlled final state at time T : $y(T)$ on the left columns. The test case corresponds to the control problem \mathcal{P}_1^α , where α is taken as $\alpha = 1 \times 10^{-02}$.

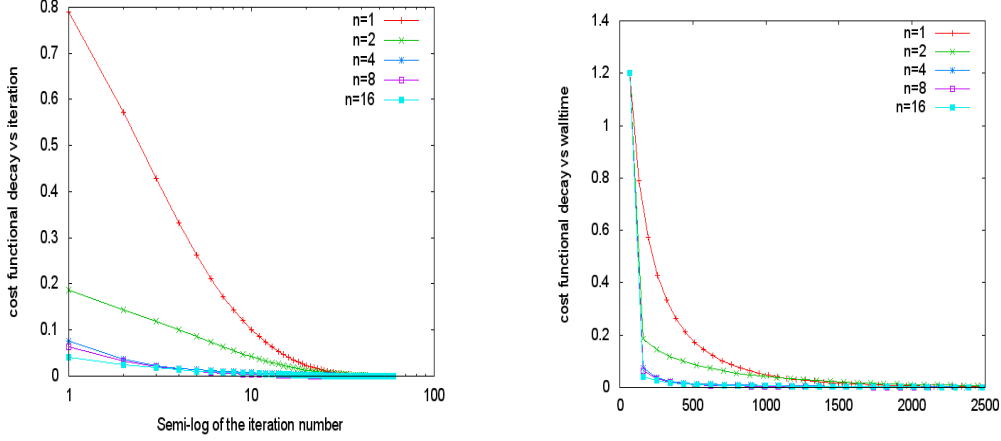


Figure 5: First test problem, for \mathcal{P}_2^α : Normalized cost functional values versus time of computation for several values of \hat{n} (i.e. the number of processors used).

respectively.

Remark 5.1. *Because of the linearity and the superposition property of the heat equation, it can be shown that problems (\mathcal{P}_2^α) and \mathcal{P}_3^α) mentioned above are equivalent to a control problem which has null target solution.*

Test problem	Results				
\mathcal{P}_1^α	$\alpha = 1 \times 10^{-02}$				
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$
	Number of iterations k	100	68	63	49
	walltime in mn	15311.6	15352.3	14308.7	10998.2
	$\ y^k(T) - y^{target}\ _2 / \ y^{target}\ _2$	0.472113	0.472117	0.472111	0.472104
\mathcal{P}_2^α	$\alpha = 1 \times 10^{-02}$				
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$
	Number of iterations k	60	50	45	40
	walltime in mn	3855.21	3726.28	4220.92	3778.13
	$\ y^k(T) - y^{target}\ _2 / \ y^{target}\ _2$	8.26×10^{-08}	8.26×10^{-08}	8.15×10^{-08}	8.15×10^{-08}
\mathcal{P}_2^α	$\alpha = 1 \times 10^{-08}$				
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$
	Number of iterations k	60	50	40	30
	walltime in mn	3846.23	4654.34	3759.98	2835.31
	$\ y^k(T) - y^{target}\ _2 / \ y^{target}\ _2$	3.93×10^{-08}	1.14×10^{-08}	5.87×10^{-09}	2.04×10^{-09}

Table 1: Results' summary of **Algorithm 3** applied on the distributed control problems \mathcal{P}_1^α and \mathcal{P}_2^α .

5.2.2. Second test problem

In this section, we are concerned with the approximate controllability of the heat equation, where the regularization parameter α vanishes, practically we take $\alpha = 1 \times 10^{-08}$. In this case, problems \mathcal{P}_2^α and \mathcal{P}_3^α , in the continuous setting are supposed to be well posed (see .e.g [15, 16]

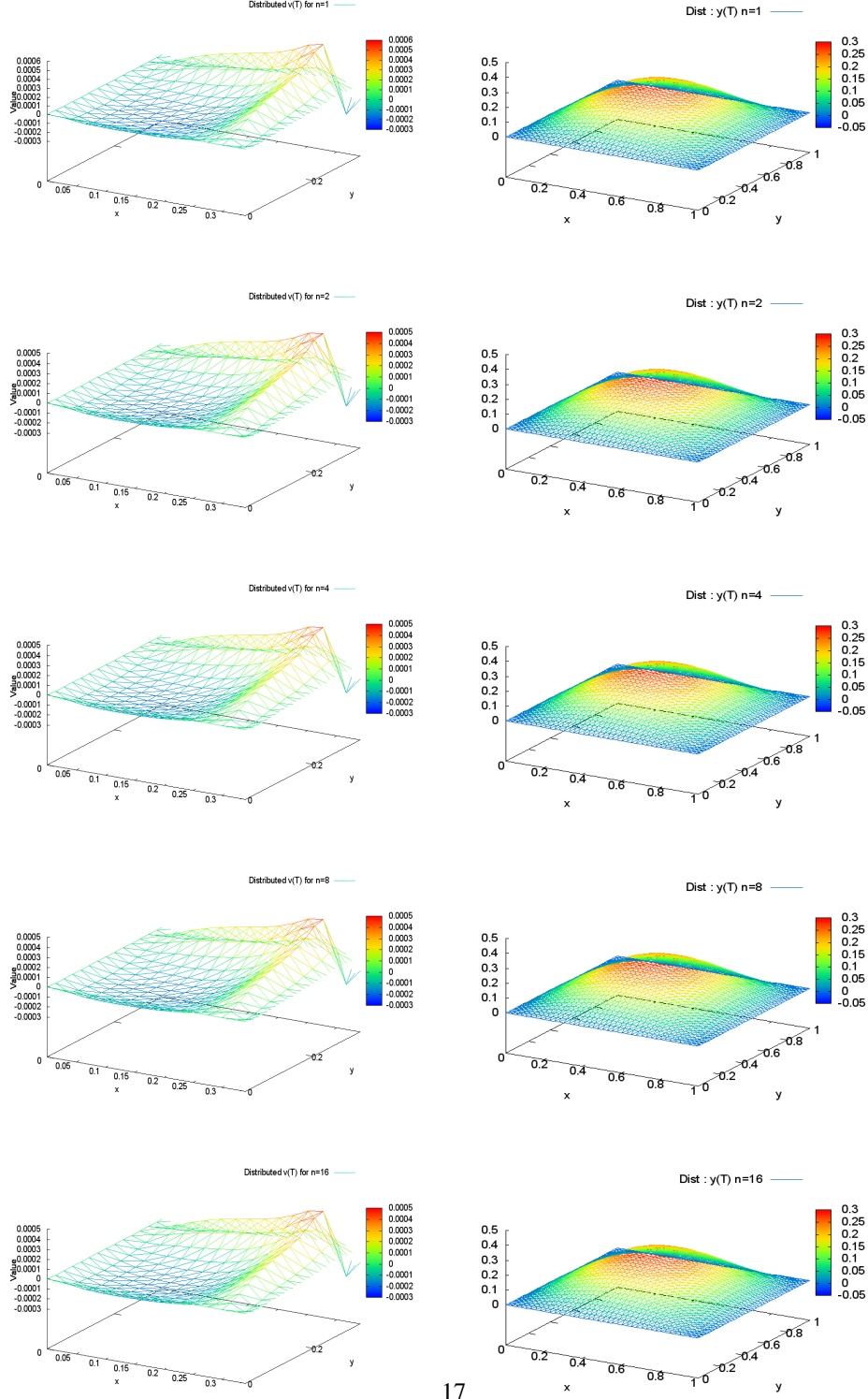


Figure 6: Several rows value snapshots in \hat{n} of the distributed optimal control on the left columns and its corresponding controlled final state at time T: $y(T)$ on the right columns. The test case corresponds to the control problem \mathcal{P}_2^α , where $\alpha = 1 \times 10^{-02}$.

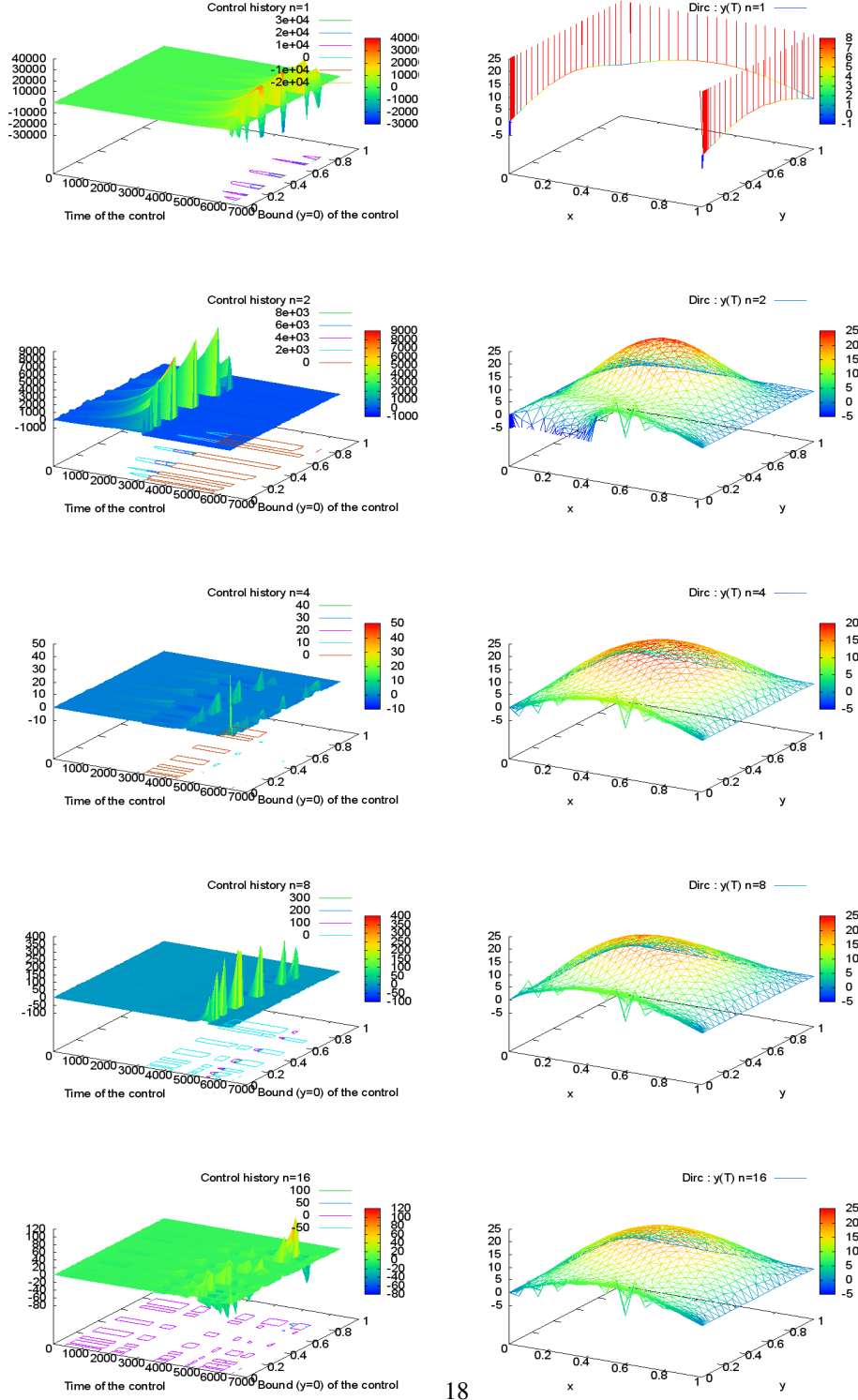


Figure 7: Several rows value snapshots in \hat{n} of the Dirichlet optimal control on the left columns and its corresponding controlled final state at time T : $y(T)$ on the right columns. The test case corresponds to the control problem \mathcal{P}_3^α , where $\alpha = 1 \times 10^{-02}$.

and references therein). However, may not be the case in the discretized settings; we refer for instance to [12] (and reference therein) for more details.

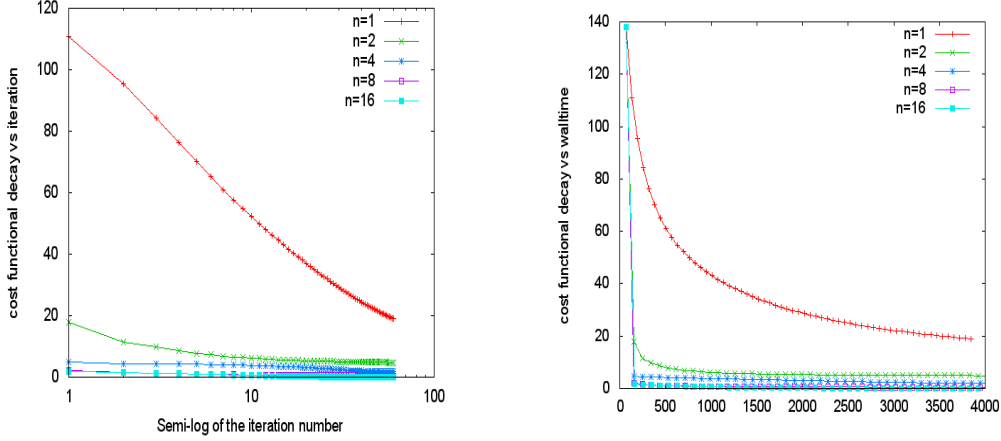


Figure 8: Normalized and shifted cost functional values versus time of computation for several values of \hat{n} (i.e. the number of processors used), Distributed control problem \mathcal{P}_2^α with $\alpha = 1 \times 10^{-08}$.

Test problem		Results				
\mathcal{P}_3^α	$\alpha = 1 \times 10^{-02}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	40	40	30	18	10
	walltime in mn	12453.9	12416.1	9184.28	5570.54	3158.97
	$\ y(T) - y^{target}\ _2 / \ y^{target}\ _2$	$8.54 \times 10^{+06}$	0.472488	0.0538509	0.0533826	0.0534024
\mathcal{P}_3^α	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	40	40	30	27	10
	walltime in mn	1248.85	1248.97	916.232	825.791	325.16
	$\ y(T) - y^{target}\ _2 / \ y^{target}\ _2$	$8.85 \times 10^{+06}$	0.151086	0.0292072	0.0278316	0.0267375
	$\int_{(0,T)} \ v\ _1^2 dt$	$7.92 \times 10^{+08}$	$2.30 \times 10^{+07}$	$1.27 \times 10^{+07}$	$1.47 \times 10^{+07}$	$1.58 \times 10^{+06}$

Table 2: Results' summary of **Algorithm 3** applied on the Dirichlet boundary control problems \mathcal{P}_2^α and \mathcal{P}_3^α .

Table 1 contains the summarized results for the convergence of the distributed control problem. On the one hand, we are interested in the error given by our algorithm for several choices of partition number \hat{n} . On the other hand, we give the $L^2(0, T; L^2(\Omega_c))$ of the control.

We notice the improvement in the quality of the algorithm in terms of both time of execution and control energy consumption. In fact, for the optimal control framework ($\alpha = 1 \times 10^{-02}$ relatively big), we see that, for a fixed threshold, the algorithm is faster and consume less energy as \hat{n} increases.

For the approximate controllability framework ($\alpha = 1 \times 10^{-08}$ vanishes), we note first that the general accuracy of the controlled solution is improved as α diminishes. Second, we note that the error diminishes with respect to increasing \hat{n} ; however the energy consumption rises as well. The scalability in time and number of iteration is not directly affected by the change of the problem in α .

Table 2 contains the summarized results at the convergence of the Dirichlet boundary control

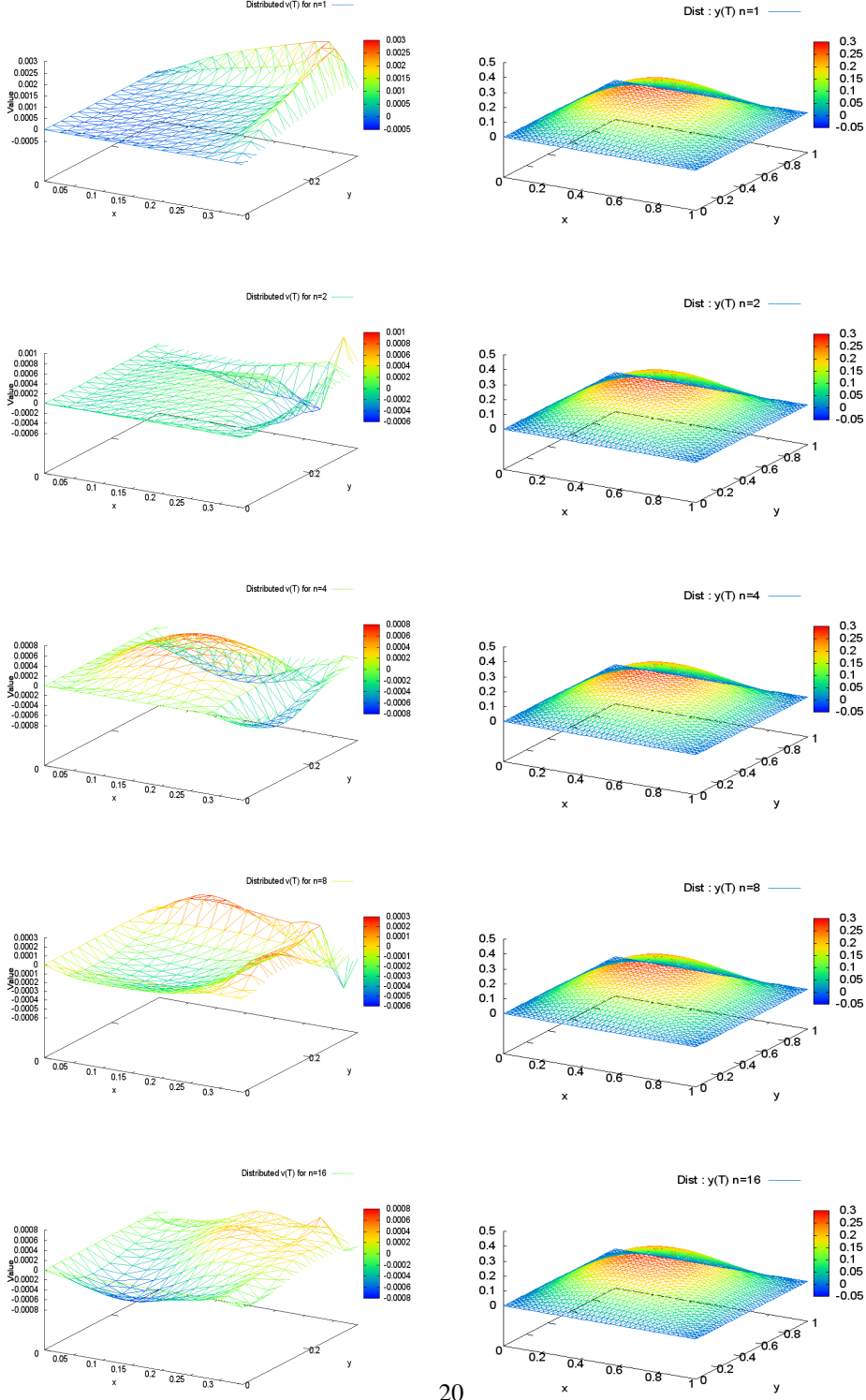


Figure 9: Several rows value snapshots in \hat{n} of the distributed optimal control on the left columns and its corresponding controlled final state at time T : $y(T)$ on the right columns. The test case corresponds to the control problem \mathcal{P}_2^α , where $\alpha = 1 \times 10^{-08}$.

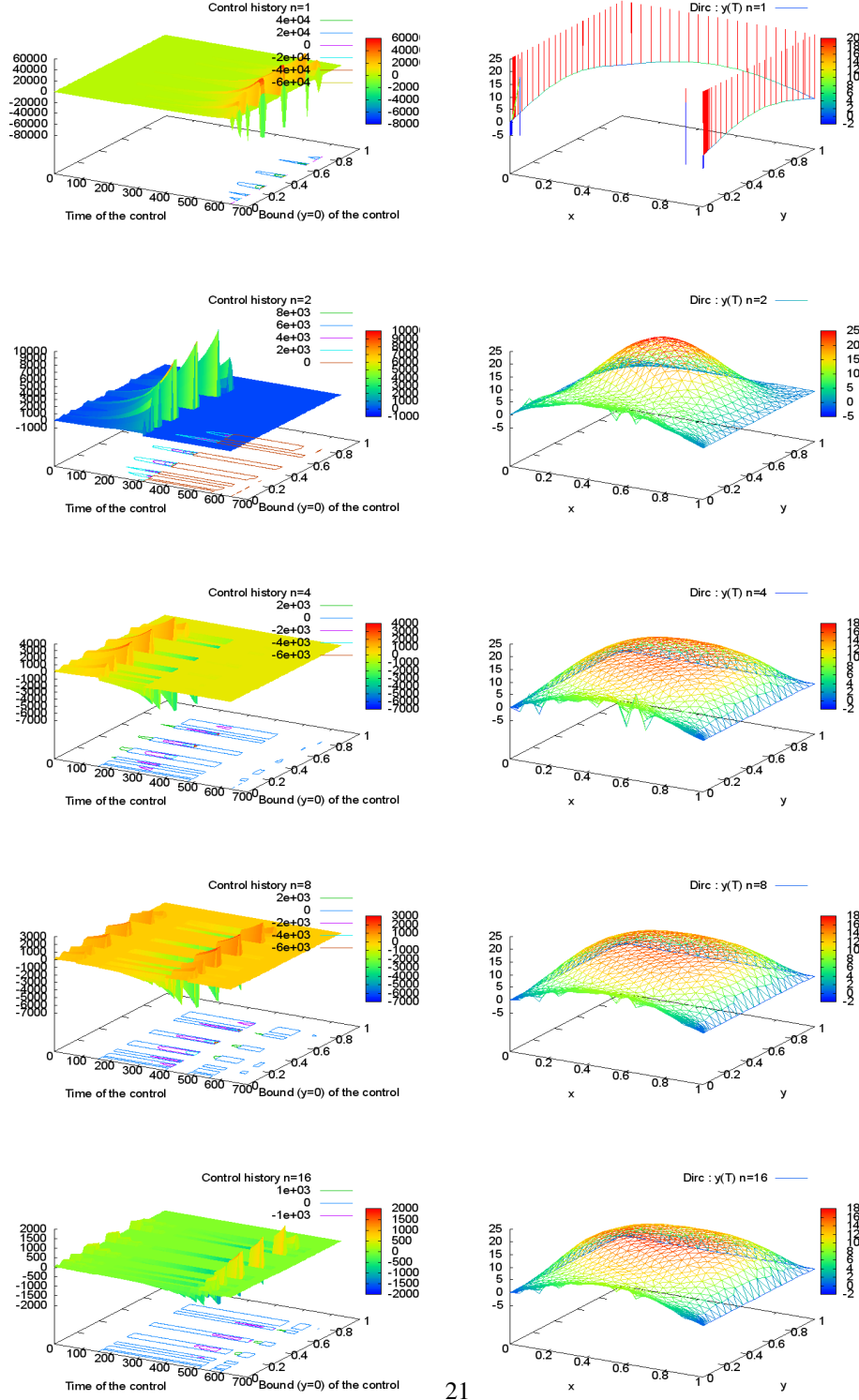


Figure 10: Several rows value snapshots in \hat{n} of the Dirichlet optimal control on the left columns and its corresponding controlled final state at time T : $y(T)$ on the right columns. The test case corresponds to the control problem \mathcal{P}_3^α , where $\alpha = 1 \times 10^{-08}$.

problem. This problem is known in the literature for its ill-posedness, where it may be singular in several cases [11]. In fact, it is very sensitive to noise in the data. We show in Table 2 that for a big value of the regularization parameter α our algorithm behaves already as the distributed optimal control for a vanishing α , in the sense that it consumes more control energy to produce a more accurate solution with smaller execution time. It is worth noting that the serial case $\hat{n} = 1$ fails to converge, whereas the algorithm behaves well as \hat{n} rises.

We give in Fig. 7 and Fig. 10 several rows value snapshots (varying \hat{n}) of the Dirichlet control on Γ . We present in the first column its evolution during $[0, T]$ and on the second column its corresponding controlled final solution $y(T)$ at time T ; we scaled the plot of the z-range of the target solution in both Figs. 7 and 10.

In each row one sees the control and its solution for a specific partition \hat{n} . We notice that the serial case $\hat{n} = 1$ fails to reach the target solution even if we have a stable minimization procedure of the cost function.

The serial case $\hat{n} = 1$ leads to a controlled solution which does not have the same rank as y^{target} , whereas as \hat{n} rises, we improve the results. It is worth noting that the control is generally active only around the final horizon time T . This is very clear in Fig. 7 and Fig. 10 (see the first row i.e. case $\hat{n} = 1$). The nature of our algorithm, which is based on time domain decomposition, obliges the control to act in subintervals. Hence, the control acts more often and earlier in time (before T) and leads to a better controlled solution $y(T)$.

5.2.3. Third test problem

In this test case, we consider a severely ill-posed problem. In fact, the target solution is piecewise Lipschitz continuous, so that it is not regular enough compared with the solution of the heat equation. This implies that in our control problem, both the distributed and the Dirichlet boundary control has no solution. The initial condition and the target solution are given by

$$\begin{aligned} y_0(x_1, x_2) &= \pi(\sin(\pi x_1) + \sin(\pi x_2)) \\ y^{target}(x_1, x_2) &= \min(x_1, x_2, (1 - x_1), (1 - x_2)), \end{aligned} \quad (\mathcal{P}_4^\alpha)$$

respectively. A plots of the initial condition and the target solutions are given in Fig. 11.

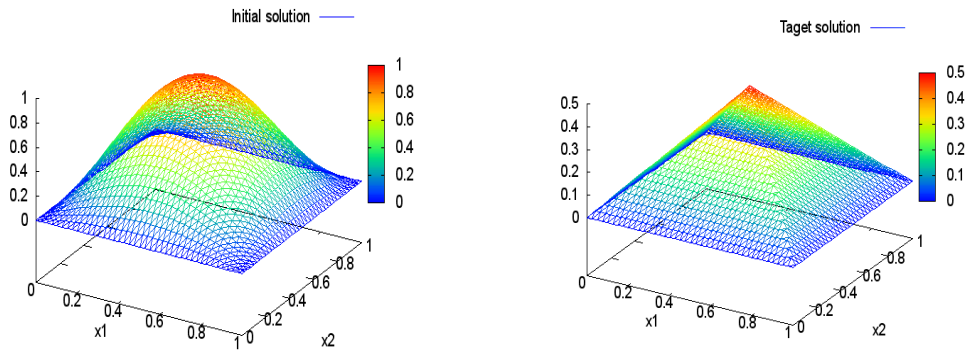


Figure 11: Graph of initial and target solution for both distributed and Dirichlet boundary control problem.

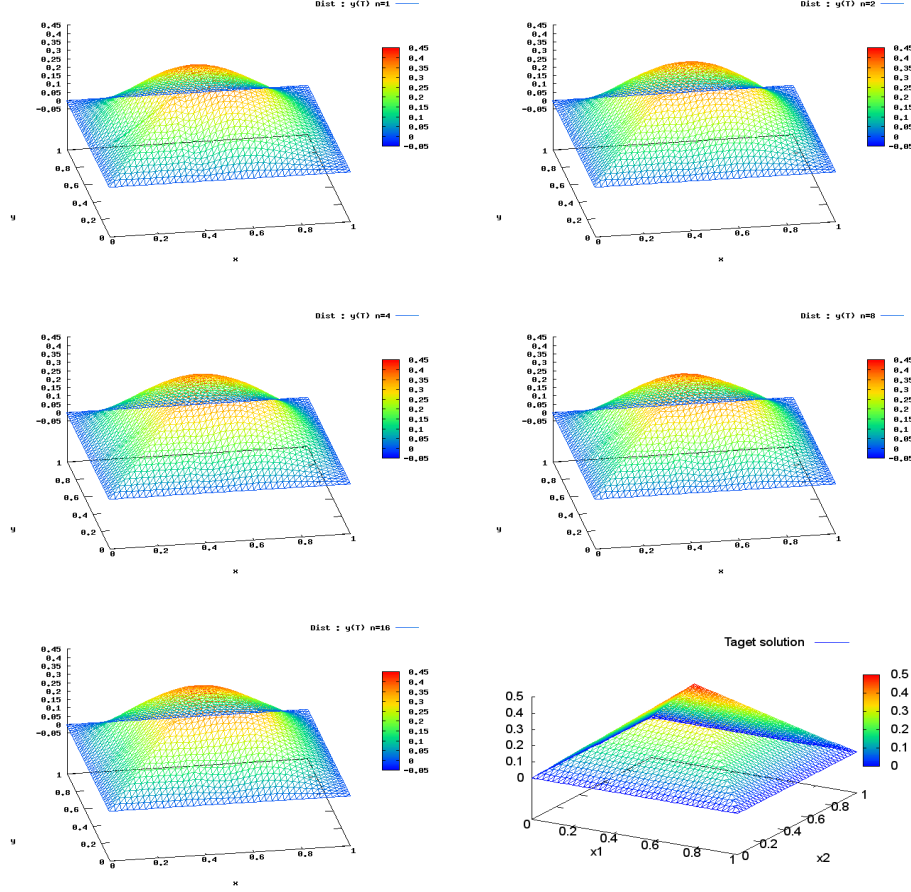


Figure 12: Several snapshots in \hat{n} of final state at time T: $y(T)$. The test case corresponds to Distributed control sever Ill-posed problem \mathcal{P}_4^α .

Test problem		Results				
Distributed control \mathcal{P}_4^α	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	100	68	60	50	40
	walltime in mn	6381.43	6303.67	5548.16	4676.83	3785.97
	$\ y(T) - y^{target}\ _2 / \ y^{target}\ _2$	8.16×10^{-03}	5.3×10^{-03}	4.74×10^{-03}	3.95×10^{-03}	3.76×10^{-03}
Dirichlet control \mathcal{P}_4^α	$\alpha = 1 \times 10^{-08}$					
	Quantity	$\hat{n} = 1$	$\hat{n} = 2$	$\hat{n} = 4$	$\hat{n} = 8$	$\hat{n} = 16$
	Number of iterations	25	25	20	4	1
	walltime in mn	848.58	655.40	655.40	146.19	62.87
	$\ y(T) - y^{target}\ _2 / \ y^{target}\ _2$	$2.85 \times 10^{+10}$	3055	39.3	0.2	0.067
	$\int_{(0,T)} \ v\ _T^2 dt$	$6.73 \times 10^{+08}$	$2.17 \times 10^{+07}$	141.62	17.84	26758.5

Table 3: Results' summary of **Algorithm 3** applied on to both distributed and Dirichlet boundary control for the third test problem \mathcal{P}_4^α .

In Figures 12 and 13 we plot the controlled solution at time T for the distributed and Dirichlet control problems respectively. We remark that for the distributed control problem the controlled solution is smooth except in Ω_c , where the control is able to fit with the target solution.

Remark 5.2. *Out of curiosity, we tested the case where the control is distributed on the whole domain. We see that the control succeeds to fit the controlled solution to the target even if it is discontinuous. This is impressive and shows the impact on the results of the regions where the control is distributed.*

We note the stability of the method of the distributed test case. However, the Dirichlet problem test case presents hypersensitivity. In fact, in the case of $\hat{n} = 1$ the algorithm succeeds to fit an acceptable shape of the controlled solution, although still far in the scale. We note that the time domain decomposition leads to a control which gives a good scale of the controlled solution. In this severely ill-posed problem, we see that some partitions may fail to produce a control that fits the controlled solution to the target. There is an exemption for the case of $\hat{n} = 8$ partitions, where we have a good reconstruction of the target. The summarized results are given in Tables 3.

5.2.4. Regularization based on the optimal choice of partition

The next discussion concerns the kind of situation where the partition leads to multiple solutions, which is common in ill-posed problems. In fact, we discuss a regularization procedure used as an exception handling tool to choose the best partition, giving the best solution of the handled control problem.

It is well known that ill-posed problems are very sensitive to noise, which could be present due to numerical approximation or to physical phenomena. In that case, numerical algorithm may blow-up and fail. We present several numerical tests for the Dirichlet boundary control, which is a non trivial problem numerically. The results show that in general time domain decomposition may improve the results in several cases. But scalability is not guaranteed as it is for the distributed control (see for instance Fig. 14). We propose a regularization procedure in order to avoid the blow-up and also to guarantee the optimal choice of partition of the time domain. This procedure is based on a test of the monotony of the cost function. In fact, suppose that we possess 64 processors to run the numerical problem. Once we have assembled the Hessian H_k and the Jacobian D_k for the partition $\hat{n} = 64$, we are actually able to get for free the results of the Hessian and the Jacobian for all partitions \hat{n} that divide 64. Hence, we can use the quadratic property of the cost functional in order to predict and test the value of the cost function for the next iteration without making any additional computations. The formulae is given by:

$$J(v^{k+1}) = J(v^k) - \frac{1}{2} D_k^T H_k^{-1} D_k.$$

We present in **Algorithm 4** the technique that enables us to reduce in rank and compute a series of Hessians and Jacobians for any partition \hat{n} that divide the available number of processors. An exemple of the applicability of these technique, on a 4-by-4 SPD matrix, is given in Appendix.

6. Conclusion

We have presented in this article a time parallel approach for the simulation of optimal control problem for systems governed by PDEs. The method is based on the calculation of the vector step-length according to a set of gradient descent directions using a quasi-Newton technique.

Algorithm 4: Reduce in rank of the partition \hat{n}

```

0 Input:  $\hat{n}, H_{\hat{n}}^k, D_{\hat{n}}^k$ ;
1  $n = \hat{n}$ ;
2  $J_{n/2}^{k+1} = J_n^{k+1}$ ;
3 while  $J_{n/2}^{k+1} > J_n^k$  do
4   for  $i = 0; i \leq n; i + 2$  do
5      $(D_{n/2}^k)_i = (D_n^k)_i + (D_n^k)_{i+1}$ ;
6   for  $j = 0; j \leq n; j + 2$  do
7      $(H_{n/2}^k)_{i,j} = (H_n^k)_j + (H_n^k)_{j+1}$ ;
8   end
9 end
10 Estimation of the cost  $J_{n/2}^k$ ;
11  $n = \frac{n}{2}$ ;
12 end

```

It is guaranteed that the new algorithm performs much better than the optimal steepest descent algorithm in the well-posed settings. However, for the ill-posed settings, we have to regularize the descent directions in order to obtain performance improvement. Convergence property of the presented method is provided for any arbitrary partition for the well-posed case. Those results are illustrated with several numerical tests using parallel resources with MPI implementation.

Kantorovich matrix inequality. For the sake of completeness, we give in this appendix the Matrix Kantorovich inequality, that justifies the statement of our convergence proof. Assume that $\nabla^2 J$ is symmetric positive definite with smallest and largest eigenvalues λ_{\min} and λ_{\max} respectively. We give in the following the matrix version of the famous Kantorovich inequality, which reads:

Theorem 6.1 (see [7] for more details). *Assume that $\sum_{n=1}^{\hat{n}} \alpha_n = 1$ where $\alpha_n \geq 0$ and $\lambda_n > 0 \quad \forall n$; we have thus :*

$$\sum_{n=1}^{\hat{n}} \alpha_n \lambda_n \sum_{n=1}^{\hat{n}} \frac{\alpha_n}{\lambda_n} \leq \frac{(\lambda_{\max} + \lambda_{\min})^2}{4\lambda_{\max}\lambda_{\min}}.$$

By diagonalizing the symmetric positive definite operator H we obtain: $H = P\Lambda P^{-1}$, where P is orthonormal operator (i.e. $P^T = P^{-1}$). Recall Eq.(33) that we rewrite as:

$$\frac{\|\nabla J(v^k)\|_{\nabla^2 J}^2 \|\nabla J(v^k)\|_{(\nabla^2 J)^{-1}}^2}{\|\nabla J(v^k)\|_c^4} \leq \frac{(\lambda_{\max} + \lambda_{\min})^2}{4\lambda_{\max}\lambda_{\min}}.$$

In order to simplify the expression, we shall use d_k instead of $\nabla J(v^k)$ so that the equation above reads:

$$\frac{d_k^T (\nabla^2 J) d_k \, d_k^T (\nabla^2 J)^{-1} d_k}{(d_k^T d_k)^2} = \frac{d_k^T P^T \Lambda P d_k \, d_k^T P^T \Lambda^{-1} P d_k}{d_k^T P^T P d_k \, d_k^T P^T P d_k}.$$

Let us define $\mathbf{d}_k := P d_k$, consequently the above equality becomes:

$$\frac{\mathbf{d}_k^T \Lambda \mathbf{d}_k \, \mathbf{d}_k^T \Lambda^{-1} \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{d}_k \, \mathbf{d}_k^T \mathbf{d}_k} = \sum_{n=1}^{\hat{n}} \frac{(\mathbf{d}_k)_n^2}{\mathbf{d}_k^T \mathbf{d}_k} \lambda_n \sum_{n=1}^{\hat{n}} \frac{(\mathbf{d}_k)_n^2}{\mathbf{d}_k^T \mathbf{d}_k} \frac{1}{\lambda_n}.$$

We then denote by $\alpha_n = \frac{(\mathbf{d}_k)_n^2}{\mathbf{d}_k^T \mathbf{d}_k}$ so that $\sum_{n=1}^{\hat{n}} \alpha_n = 1$, and finally:

$$\frac{d_k^T A d_k d_k^T A^{-1} d_k}{(d_k^T d_k)^2} = \sum_{n=1}^{\hat{n}} \alpha_n \lambda_n \sum_{n=1}^{\hat{n}} \frac{\alpha_n}{\lambda_n}.$$

□

Exemple 4-by-4 SPD matrix reduced in rank using Algorithm 4. In order to illustrate the instance of Algorithm 4. We choose a simple example: a matrix 4-by-4 which we are going to reduce recursively in 2-by-2 and in 1-by-1 as follows:

$$\begin{pmatrix} 6 & 1 & 2 & 3 \\ 1 & 8 & 2 & 4 \\ 2 & 2 & 12 & 7 \\ 3 & 4 & 7 & 16 \end{pmatrix} \mapsto \begin{pmatrix} (6 & 1) & (2 & 3) \\ (1 & 8) & (2 & 4) \\ (2 & 2) & (12 & 7) \\ (3 & 4) & (7 & 16) \end{pmatrix} \mapsto \begin{pmatrix} 7 & 5 \\ 9 & 6 \\ 4 & 19 \\ 7 & 23 \end{pmatrix} \mapsto \begin{pmatrix} 16 & 11 \\ 11 & 42 \end{pmatrix} \\ \begin{pmatrix} 16 & 11 \\ 11 & 42 \end{pmatrix} \mapsto \begin{pmatrix} 27 \\ 53 \end{pmatrix} \mapsto (80)$$

References

- [1] R. H. Byrd, G. Lopez-Calva, J. Nocedal, [A line search exact penalty method using steering rules](#), Math. Program. 133 (1-2, Ser. A) (2012) 39–73. doi:10.1007/s10107-010-0408-0. URL <http://dx.doi.org/10.1007/s10107-010-0408-0>
- [2] L. Grippo, F. Lampariello, S. Lucidi, A nonmonotone line search technique for newton’s method, SIAM Journal on Numerical Analysis (23) (1986) 707–716.
- [3] G. Yuan, Z. Wei, [The Barzilai and Borwein gradient method with nonmonotone line search for nonsmooth convex optimization problems](#), Math. Model. Anal. 17 (2) (2012) 203–216. doi:10.3846/13926292.2012.661375. URL <http://dx.doi.org/10.3846/13926292.2012.661375>
- [4] A.-L. Cauchy, Méthode générale pour la résolution des systèmes d’équations simultanées, Compte Rendu des Scieances de L’Académie des Sciences XXV S’erie A (25) (1847) 536–538.
- [5] J. Barzilai, J. M. Borwein, Two point step size gradient methods, IMA Journal of Numerical Analysis (8).
- [6] P. G. Ciarlet, Introduction à l’analyse numérique matricielle et à l’optimisation, Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master’s Degree], Masson, Paris, 1982.
- [7] L. V. Kantorovich, Functional analysis and applied mathematics, NBS Rep. 1509, U. S. Department of Commerce National Bureau of Standards, Los Angeles, Calif., 1952, translated by C. D. Benster.
- [8] J. Baksalary, S. Puntanen, [Generalized matrix versions of the cauchy-schwarz and kantorovich inequalities](#), aequationes mathematicae 40 (1990) 316–316. doi:10.1007/BF02112308. URL <http://dx.doi.org/10.1007/BF02112308>
- [9] Scilab Enterprises, [Scilab: Le logiciel open source gratuit de calcul numrique](#), Scilab Enterprises, Orsay, France (2012). URL <http://www.scilab.org>
- [10] C. Carthel, R. Glowinski, J.-L. Lions, [On exact and approximate boundary controllabilities for the heat equation: a numerical approach](#), J. Optim. Theory Appl. 82 (3) (1994) 429–484. doi:10.1007/BF02192213. URL <http://dx.doi.org/10.1007/BF02192213>
- [11] F. Ben Belgacem, S. M. Kaber, [On the Dirichlet boundary controllability of the one-dimensional heat equation: semi-analytical calculations and ill-posedness degree](#), Inverse Problems 27 (5) (2011) 055012, 19. doi:10.1088/0266-5611/27/5/055012. URL <http://dx.doi.org/10.1088/0266-5611/27/5/055012>
- [12] S. Ervedoza, E. Zuazua, [The wave equation: Control and numerics](#) (2012) 245–339doi:10.1007/978-3-642-27893-8_5. URL http://dx.doi.org/10.1007/978-3-642-27893-8_5
- [13] O. Pironneau, F. Hecht, J. Morice, freeFEM++, www.freefem.org/.

- [14] J.-L. Lions, Optimal control of systems governed by partial differential equations., Translated from the French by S. K. Mitter. Die Grundlehren der mathematischen Wissenschaften, Band 170, Springer-Verlag, New York, 1971.
- [15] J.-M. Coron, E. Trélat, [Global steady-state controllability of one-dimensional semilinear heat equations](#), SIAM J. Control Optim. 43 (2) (2004) 549–569. doi:10.1137/S036301290342471X.
URL <http://dx.doi.org/10.1137/S036301290342471X>
- [16] S. Micu, E. Zuazua, [Regularity issues for the null-controllability of the linear 1-d heat equation](#), Systems Control Lett. 60 (6) (2011) 406–413. doi:10.1016/j.sysconle.2011.03.005.
URL <http://dx.doi.org/10.1016/j.sysconle.2011.03.005>